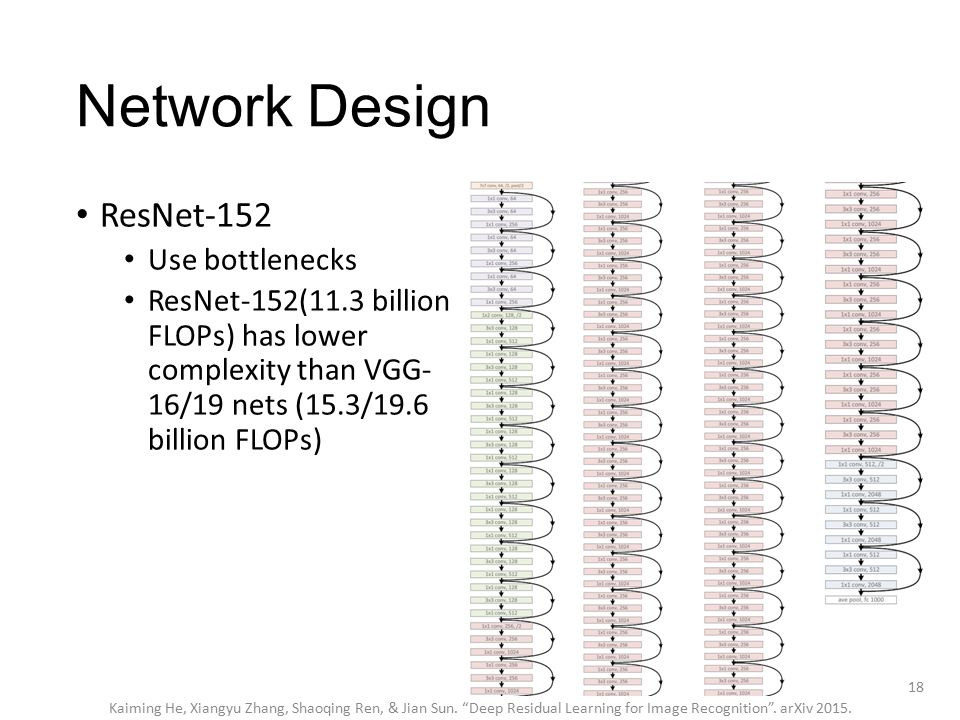
**Small Documentation on procedure**

* Prepare a Dataset, We need a folder of Images to analyze, Mounting the Google drive to axcess the Image Data.
* We will load a previously-trained neural network, that of Resnet152, which comes with Keras. If it's your first time loading it, it will automatically download the weights for you, which will take some time. Afterwards, the weights will be cached locally for future use.
* Architecture of ResNet-152
* 
* Forwarding an image through Network.
* Let's get class predictions from this model. We forward x through model and then use the built-in decode\_predictions to look up the class names, We can see here that predicted tiger with probability 0.915
* The next step will instantiate a PCA object, which we will then fit our data to, choosing to keep the top 300 principal components. This may take a few minutes.
* We are now ready to do our reverse image queries! The matrix pca\_features contains a compact representation of our images, one 300-element row for each image with high-level feature detections. We should expect that two similar images, which have similar content in them, should have similar arrays in pca\_features. Let's pick a random Input image.
* The assumption we can now make is that two images which have similar content, should produce similar feature vectors. The next two cells will randomly select one image, and then compute a measurement of the dissimilarity (or distance) of that image's PCA feature vector to every other image's feature vector. The dissimilarity metric we use is cosine distance.
* For convenience, we'll automate this process by defining the function get\_closest\_images, which will compute the cosine distance between the PCA features of query\_image\_idx-th image in our dataset, and the PCA features of every image in the dataset (including itself, trivially 0). It then returns an array of indices to the num\_results (default is 5) most similar images to it (not including itself).
* We also define a helper function get\_concatenated\_images which will simply take those resulting images and concatenate them into a single image for easy display.
* So now we can do our lookups and final results.